

ฟังก์ชัน (Function)

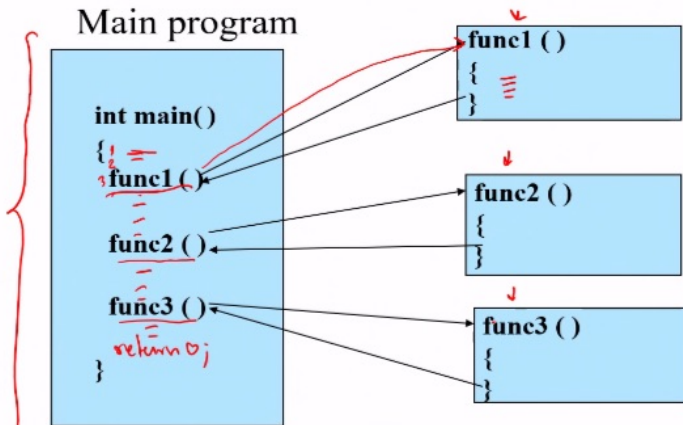
การออกแบบโปรแกรมในภาษาซีจะอยู่บนพื้นฐานของการออกแบบโมดูล (Module Design) โดยการแบ่งโปรแกรมออกเป็นงานย่อย ๆ (หรือโมดูล) แต่ละงานย่อยจะทำงานอย่างใดอย่างหนึ่งเท่านั้น และไม่ควรมีขนาดใหญ่จนเกินไป งานย่อยเหล่านี้เมื่อนำไปเขียนโปรแกรมในภาษาซีจะเป็นการเขียนในลักษณะของฟังก์ชัน

26 March 2022

2

ฟังก์ชัน

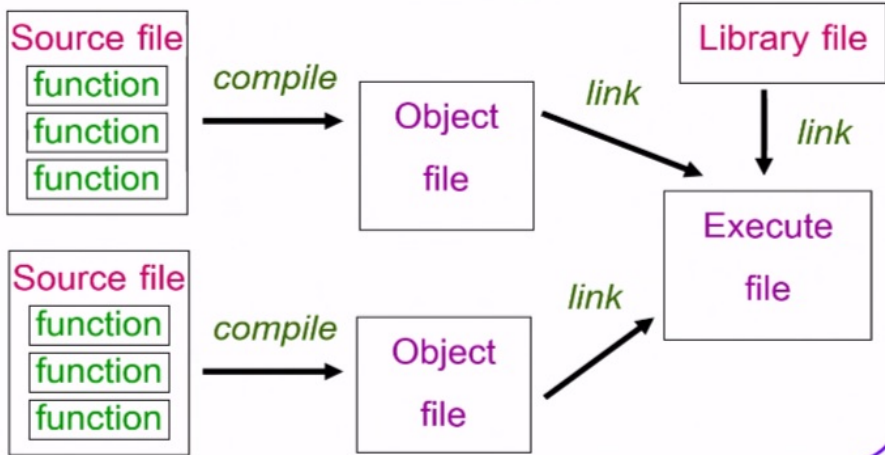
Main program



26 March 2022

4

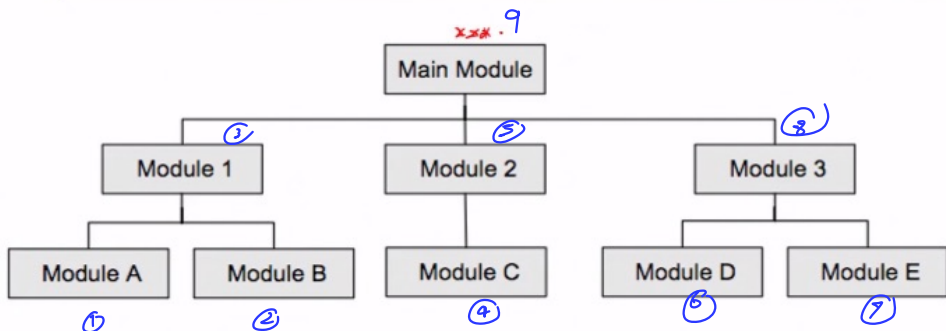
ขั้นตอนการสร้างโปรแกรมด้วยภาษา C



26 March 2022

5

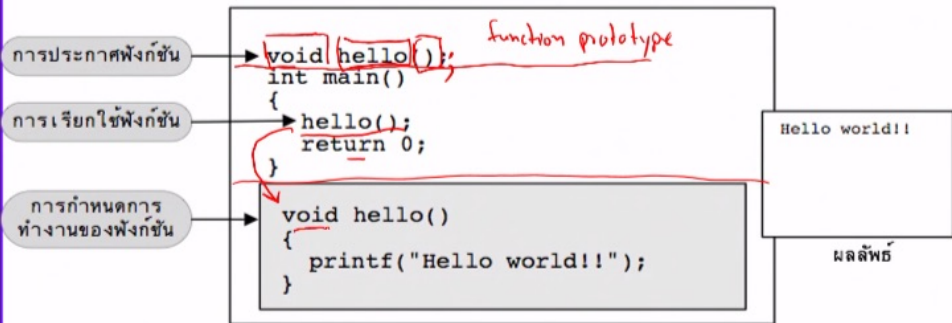
ฟังก์ชัน



26 March 2022

6

ฟังก์ชันในภาษา C: การใช้งาน



26 March 2022

9

การประกาศโปรโตไทป์ (prototype) ของฟังก์ชัน

การประกาศโปรโตไทป์เป็นสิ่งจำเป็นในภาษาซีเนื่องจากภาษาซีเป็นภาษาในลักษณะที่ต้องมีการประกาศฟังก์ชันก่อนจะเรียกใช้ฟังก์ชันนั้น (**Predefined Function**)

26 March 2022

10

โพรโทไทป์ (prototype)

ส่วนประกอบหลัก 3 ส่วนของ function header

`return type function_name (param 1, ..., param n);`

↑
ชนิดของข้อมูลที่ส่งกลับ (function return type)

↑
ชื่อฟังก์ชัน (function name)

↑
พารามิเตอร์ (parameter list)

26 March 2022

12

`int main()
return 0;
(n)`

- ออกสอบ 1 ข้อ

ตัวอย่าง 2 (ต่อ)

`int
void main()`

{

`double a1, a2, sumVal;`

→ `a1 = InputDouble();`

→ `a2 = InputDouble();`

→ `sumVal = SumDouble(a1, a2);`

`PrintOut(sumVal);`

`return 0;`

}

26 March 2022

14

ตัวอย่าง 2

แสดงการทำงานของโปรแกรมการบวกเลขจำนวน
จริง 2 จำนวนที่รับจากผู้ใช้ ในลักษณะที่มีการ
ประกาศโปรโตไทป์

```
#include <stdio.h>
double InputDouble();
double SumDouble(double, double);
void PrintOut(double);
```

26 March 2022

13

องค์ประกอบของฟังก์ชัน

Function Header

return_type function_name (parameter_list)

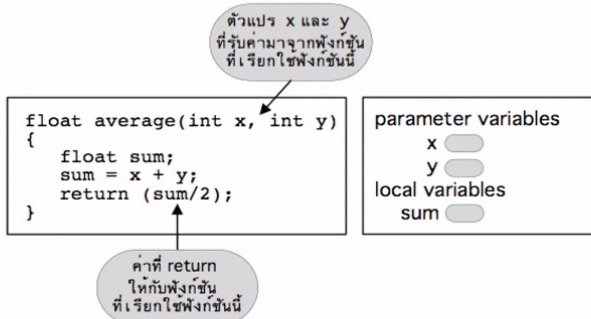
Function Body

```
{  
    // Local Declarations ประกาศตัวแปรในฟังก์ชัน  
    ...  
    // Statements  
    ...  
    // Return  
}
```

26 March 2022

16

Parameter และ Local variable



26 March 2022

20

ตัวอย่าง

a1 และ a2 ต้องมีชนิดเป็น double
เพื่อให้ตรงกับชนิดตัวแปรของอาร์กิวเมนต์
ที่ประกาศในโปรโตไทป์

```
sumVal = SumDouble(a1,a2);  
        ใช้คู่กับโปรโตไทป์  
double SumDouble(double, double);
```

26 March 2022

23

การออกแบบฟังก์ชัน

- วิเคราะห์การทำงานของโปรแกรม
- แบ่งการทำงานออกเป็นส่วนย่อยที่ทำงานอย่างใดอย่างหนึ่ง (งานส่วนย่อยไม่ควรมีขนาดใหญ่เกินไป)

26 March 2022

27

ฟังก์ชันที่ไม่มีการรับค่าและส่งค่ากลับ

The diagram illustrates the relationship between a function prototype and its definition. It consists of two code snippets with numbered lines and arrows indicating the flow of execution.

Function Prototype:

```
1 void hello();
```

Function Definition:

```
2 int main()  
3 {  
4     printf("Function main 1\n");  
5     hello();  
6     printf("Function main 2\n");  
7     return 0;  
8 }  
9 void hello()  
10 {  
11     printf("Hello world!!\n");  
12 }
```

Call Flow:

- Line 4 calls `hello()`, which jumps to line 9.
- Line 6 calls `printf`, which jumps to line 3.
- Line 9 is the start of the `hello` function, which jumps back to line 5.

Annotations:

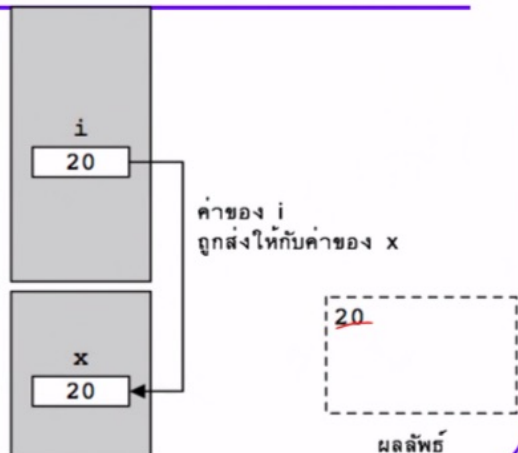
- A red box highlights `void hello();` with the handwritten text "function prototype" above it.
- A red box highlights `hello();` with the number "2" in a circle next to it.
- A red box highlights `void hello()` with the number "1" in a circle next to it.
- A red box highlights the body of the `hello` function (lines 10-12) with the text "จบการทำงาน" (End of work) below it.

```
Function main 1
Hello World
Function main 2
```

ผลลัพธ์

ฟังก์ชันที่มีการรับค่าแต่ไม่มีการส่งค่ากลับ

```
1 void showValue(int x);
2 int main()
3 {
4     int i = 20;
5     showValue(i);
6     return 0;
7 }
8 void showValue(int x)
9 {
10     printf("%d\n", x);
11 }
```



ฟังก์ชันที่ไม่มีการรับค่าแต่มีการส่งค่ากลับ

```

1 int getInput();
2 int main()
3 {
4   → int input;
5   input = getInput();
6   printf("input = %d", input);
7   return 0;
8 }
9 int getInput()
10 {
11   → int num; local variable
12   printf("Input number");
13   scanf("%d", &num);
14   return num;
15 }

```

26 March 2022



ค่าที่ return ผ่านตัวแปร
num จากฟังก์ชัน getInput
ถูกกำหนดค่าให้ตัวแปร input
ในฟังก์ชัน main

Input number: 50
input = 50

ผลลัพธ์
สมมติว่ารับค่า 50

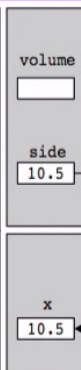
33

ฟังก์ชันที่มีการรับค่าแต่มีการส่งค่ากลับ

```

1 float cubeVol(float x);
2 int main()
3 {
4   float side = 10.5;
5   float volume;
6   volume = cubeVol(side);
7   printf("Volume of cube = %f", volume);
8   return 0;
9 }
10 float cubeVol(float x)
11 {
12   return x * x * x;
13 }

```



ค่าที่ return จาก
ฟังก์ชัน cubeVol
ถูกกำหนดให้ ตัวแปร
volume

Volume of cube = 1157.625

ผลลัพธ์

26 March 2022

34

การส่งพารามิเตอร์ให้กับฟังก์ชัน

Argument & Parameter

```
void printValue (int x, float y);
```

```
int main()
```

```
{
```

```
    int a = 10;
```

```
    ...
```

```
    > printValue (a, 20.5);
```

```
    ...
```

```
}
```

```
void printValue ((int x, float y))
```

```
{
```

```
    printf("%d %f \n", x, y);
```

```
}
```

argument
ที่ส่งให้ฟังก์ชัน
printValue

- จำนวนของ argument
ต้องเท่ากับจำนวนของ
parameter

- ชนิดข้อมูลของ argument
ต้องเหมือนกับชนิดข้อมูลของ
parameter

parameter ของ
ฟังก์ชัน printValue

26 March 2022

37

Pass by value

```
void add(int x, int y);
```

```
int main()
```

```
{
```

```
    int a = 10;
```

```
    int b = 15;
```

```
    printf("a = %d\n", a);
```

```
    add(a,b);
```

```
    printf("a = %d\n", a);
```

```
    return 0;
```

```
}
```

```
void add(int x, int y)
```

```
{
```

```
    x = x + y;
```

ค่าของ x
เปลี่ยนเป็น 25

```
    printf("x = %d\n", x);
```

```
}
```

ค่าตัวแปรในฟังก์ชัน main

a 10 b 15

pass by value

- copy ค่าของ a ให้กับ x
- copy ค่าของ b ให้กับ y

x 10 y 15

x 25

ค่าตัวแปรในฟังก์ชัน add

a = 10
x = 25
a = 10

ผลลัพธ์

26 March 2022

39

Global Variable

นอกจากนี้สามารถประกาศตัวแปรไว้ที่ภายนอกฟังก์ชัน บริเวณส่วนเริ่มของโปรแกรมจะเรียกว่า

ตัวแปรโกลบอล (Global Variable) ซึ่งเป็นตัวแปรที่สามารถเรียกใช้ที่ตำแหน่งใด ๆ ในโปรแกรมก็ได้ ยกเว้นในกรณีที่มีการประกาศตัวแปรที่มีชื่อเดียวกันตัวแปรโกลบอลภายในบล็อกหรือฟังก์ชัน

26 March 2022

47

ข้อควรระวังในการใช้งานฟังก์ชัน

- ต้องมีการประกาศฟังก์ชันก่อนการใช้งานฟังก์ชัน การส่งผ่านค่า **argument** ส่งให้ถูกต้องและห้ามส่งสลับที่กัน จะทำให้การทำงานผิดได้

26 March 2022

54

Function

```

1  #include <stdio.h>
2  void increment(int *var)
3  {
4      *var = *var+1;
5  }
6  int main()
7  {
8      int num=20;
9
10     increment(&num);
11     printf("Value of num is: %d", num);
12     return 0;
13 }

```

Handwritten notes and diagram:

Diagram illustrating memory addresses and values:

- Variable `num` at address `1002` contains value `20`.
- Variable `x` at address `1000` contains value `20`.
- Variable `p` at address `1002` contains value `1002` (address of `num`).

Print statements and their outputs:

```

printf("%d", x);      → 20
printf("%d", *p);     → 20
printf("%d", p);      → 1002

```

Handwritten notes on the code:

- Red arrow from `&num` to `num` in the `main` function.
- Red arrow from `*var` to `num` in the `increment` function.
- Red arrow from `num` to `&num` in the `main` function.

Function

```

1  #include
2  void swapnum ( int *var1, int *var2 )
3  {
4      int tempnum ;
5      tempnum = *var1 ;
6      *var1 = *var2 ;
7      *var2 = tempnum ;
8  }
9  int main( )
10 {
11     int num1 = 35, num2 = 45 ;
12     printf("Before swapping:");
13     printf("\nnum1 value is %d", num1); 35
14     printf("\nnum2 value is %d", num2); 35
15
16     /*calling swap function*/
17     swapnum( &num1, &num2 );
18
19     printf("\nAfter swapping:");
20     printf("\nnum1 value is %d", num1);
21     printf("\nnum2 value is %d", num2);
22     return 0;
23 }
24

```

Handwritten notes on the code:

- Red underlines under `&num1` and `&num2` in the `swapnum` function call.
- Red text "swap" written below the underlines.

โปรแกรมต่อไปนี้มีผลลัพธ์แสดงทางจอภาพอย่างไร

```
1  #include<stdio.h>
2  void function(int *x, *in y);
3  int main()
4  {
5      int x = 13;
6      int y = 6;
7      printf("x = %d, y = %d \n", x, y);
8      function(&x, &y);
9      printf("x = %d, y = %d \n", x, y);
10     return 0;
11 }
12
13 void function(int *x, int *y)
14 {
15     int temp;
16     if(*x > *y){
17         temp = *x;
18         *x = *y;
19         *y = temp * 2;
20     }
21 }
```

คำตอบของคุณ

ชี้ไป

ชี้ตัว

แสดงผล

~~ผล~~ $x = 13, y = 6$

$6, 26$

$x = 6, y = 26$

turn 13

$x = 6$

$y = 26$

Function

```

1  #include <stdio.h>
2  void disp( char ch)
3  {
4      printf("%c ", ch);
5  }
6
7  int main()
8  {
9      char arr[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'};
10     for (int x=0; x<10; x++)
11     {
12         /* I'm passing each element one by one using subscript*/
13         disp (arr[x]);
14     }
15
16     return 0;
17 }

```

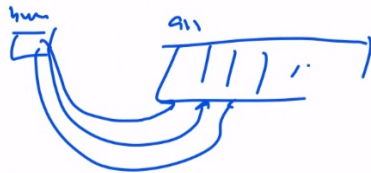
Function

```

1  #include <stdio.h>
2  void disp( int *num)
3  {
4      printf("%d ", *num);
5  }
6
7  int main()
8  {
9      int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
10     for (int i=0; i<10; i++)
11     {
12         /* Passing addresses of array elements*/
13         disp (&arr[i]);
14     }
15
16     return 0;
17 }
18

```

pass by reference
 pointer to loc



Example2.pdf x Example3.pdf x CallbyValue1 x CallbyValue2 x CallbyRefere x CallbyRefere x PassArray2f x PassArray2f x PassArray2f x

File | D:\Com_Olympiad\Function\Examples\PassArray2Function3.pdf

1 of 1

Function

```
1  #include <stdio.h>
2  void myfuncn( int *var1, int var2)
3  {
4
5      for(int x=0; x<var2; x++)
6      {
7          printf("Value of var_arr[%d] is: %d \n", x, *var1);
8          /*increment pointer for next element fetch*/
9          var1++;
10     }
11 }
12
13 int main()
14 {
15     int var_arr[] = {11, 22, 33, 44, 55, 66, 77};
16     myfuncn(var_arr, 7);
17     return 0;
18 }
```

Type here to search

11:50 26/3/2565

#include<math.h>

x

ตัวอย่างฟังก์ชัน	ความหมาย	ชนิดของตัวแปรเข้า	ชนิดของผลลัพธ์
<u>y</u> = <u>abs(x)</u>	ค่าสัมบูรณ์ของจำนวนเต็ม x	int	int
labs(x)	ค่าสัมบูรณ์ของจำนวนเต็มความสูง x	long	long
fabs(x)	ค่าสัมบูรณ์ของจำนวนจริง x	double	double
tan(x)	ค่า tangent ของ x	double	double
sin(x)	ค่า sine ของ x	double	double
cos(x)	ค่า cosine ของ x	double	double
pow(x, y)	ค่า x ยกกำลัง y	double	double
fmod(x, y)	ค่าเศษจากการหาร x/y แบบจำนวนจริง	double	double

ตัวอย่าง

```
#include<stdio.h>
#include<math.h>
#define PI 3.141592654
int main()
{
    double degree;
    double radian;
    degree = 60.0;
    radian = degree * PI/180;
    printf("degree = %f\n", degree);
    printf("radian = %f\n", radian);
    printf("cos    = %f\n", cos(radian));
    printf("sin     = %f\n", sin(radian));
    printf("tan     = %f\n", tan(radian));
    getch();
    return 0;
}
```


ตัวอย่าง

pow = 9.0000
sqrt = 4.0000
exp = 7.3891
log_e = 2.7726
Log_10 = 1.2041

```
#include<stdio.h>
#include<math.h>
```

```
int main()
```

```
{
```

```
double a = 3.0;
double b = 2.0;
double c = 16.0;
```

```
printf("%.4f\n", pow(a,b));
```

```
printf("%.4f\n", sqrt(c));
```

```
printf("%.4f\n", exp(b));
```

```
printf("%.4f\n", log(c));
```

```
printf("%.4f\n", log10(c));
```

```
getch();
```

```
return 0;
```

```
}
```

✓
→ a^b
√c
e^b
ln c log_e c
log₁₀ c

26 March 2022

69

#include<math.h>

ตัวอย่างฟังก์ชัน

ความหมาย

ชนิดของตัวแปรเข้า ชนิดของผลลัพธ์

exp(x)

คำนวณค่า e ของ x

double

double

log(x)

คำนวณค่า log ฐาน e ของ x

double

double

log10(x)

คำนวณค่า log ฐาน 10 ของ x

double

double

atof(x)

เปลี่ยนกลุ่มอักขระ x ให้เป็นจำนวนจริง

char

double

atoi(x)

เปลี่ยนกลุ่มอักขระ x ให้เป็นจำนวนเต็ม

char

integer

atol(x)

เปลี่ยนกลุ่มอักขระ x ให้เป็นจำนวนเต็มความจุสูง

char

long

ceil(x)

ค่าจำนวนเต็มที่น้อยที่สุดที่มากกว่าหรือเท่ากับ

double

double

floor(x)

ค่าจำนวนเต็มที่มากที่สุดที่น้อยกว่าหรือเท่ากับ x

double

double

sqrt(x)

ค่ารากที่สองของ x

double

double

26 March 2022

70